A B C T

rudimentary

a Bifurcation and Continuation Tool

A Project Report

AE 660 - Nonlinear Flight Mechanics

Submitted by

Ashivni Shekhawat

Abstract

ABCT is an elementary tool for limited bifurcation and continuation analysis of one parameter ODEs. ABCT provides capability for continuing fixed point curves and detection of folds, branches, and Hopf points. Two parameter continuation of fold, Hopf, branch points, and continuation of limit cycles is not implemented. All branches are computed using the pseudo arc-length continuation with tangent prediction and Newton correction. The stability of fixed points is determined using eigenvalue analysis. Being an elementary tool, ABCT is not very customizable as it stands.

1 Introduction

Bifurcation and continuation analysis are useful tools in nonlinear dynamics. Continuation techniques were originally developed for solving tough nonlinear problems. However, over the years these techniques have been applied to a very diverse set of problems, both in engineering and in pure sciences. Good introductions to this subject can be found in the works of Keller [Keller, 1987], Doedel [Doedel et al., 1991], Allgower [Allgower and Georg, 2003, 1993] and Gracia [Zangwill and Garcia, 1981] among others.

In this work we consider general one parameter ordinary differential equations of the form

$$\dot{x} \equiv \frac{dx}{dt} = f(x, \alpha), \tag{1}$$

where $x \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^1$ and $f(x, \alpha) : \mathbb{R}^{n+1} \mapsto \mathbb{R}^n$ is sufficiently smooth. Generally x represents the dynamic variables of interest and is also known as the state vector. On the other hand α represents a parameter in the system, like the damping or stiffness, and is called the parameter vector (if there are more than one parameters) or simply the parameter. The fixed points or the equilibria of the system (1) are given by

$$f(x, \alpha) = 0. \tag{2}$$

Equation 2 defines an implicit curve in $\mathbb{R}^n$. The goal of continuation algorithms is to trace out such a curve starting from one known point on the curve. The

existence of the implicit curve is ascertained by the Implicit Function Theorem under certain hypothesis. The task of continuing a curve of equilibria emanating from one equilibrium is usually accomplished by the use of *predictor-corrector* algorithms. The essence of such algorithms is to first guess or predict the next point on the curve and then refine or correct the predicted point to desired accuracy. Tangent prediction and Newton correction are amongst the most popular prediction-correction algorithms, and are used by ABCT.

As an example consider the classic 1-D normal form of the super-critical pitch fork bifurcation

$$\dot{x} = \alpha x - x^3. \tag{3}$$

This example illustrates many important aspects of continuation algorithms. The fixed points of Eq. 3 are given by

$$\alpha x - x^3 = 0. \tag{4}$$

One obvious solution to Eq. 4 is $(x, \alpha) = (0, -0.5)$. The aim of a continuation algorithm is then to start at this point and trace the entire family of solutions of Eq. 4 emanating from this point. Figure 1 shows the bifurcation diagram of Eq. 3 in the $\alpha - x$ space. Note that two fixed point curves intersect at the point $(0, 0)$. Such points are called branch points. Another task of a continuation algorithm is to identify such points and to be able to continue all branches emanating from a branch point, if needed. Further, notice that the curved branch folds (has an

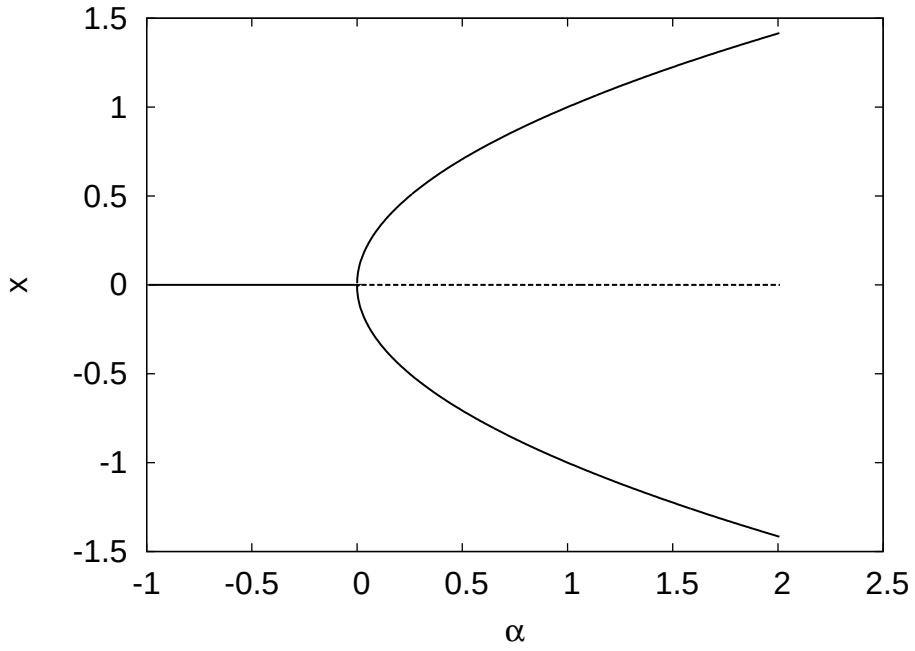

Figure 1: Bifurcation diagram showing a supercritical pitchfork bifurcation

extremum with respect to the parameter α) at the point $(0, 0)$. Such points are called fold points. Also, the straight branch changes stability as it passes through the point $(0, 0)$. Such points are called bifurcation points. Fold points are necessarily bifurcation points as well. It is the job a good continuation and bifurcation algorithm to keep track of such points. ABCT has capabilities to carry out all the above mentioned tasks, and more. The continuation algorithm used in ABCT was discussed previously a cursory manner. Next we give an overview of the branching and bifurcation detection algorithms.

Let v be the tangent vector to the fixed point curve at a point $y = (x, \alpha)$. Then

$$J(y)v = 0, \tag{5}$$

where

$$J \equiv \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial \alpha} \end{bmatrix}, \tag{6}$$

and

$$v \equiv \begin{bmatrix} \dot{x} \\ \\ \dot{\alpha} \end{bmatrix}; \quad ||v|| = 1. \tag{7}$$

It is clear from Eq. 5 that

$$v \in N(J), \tag{8}$$

where $N(J)$ is the null-space of J. A regular point of the equilibrium curve is defined as a point for with the null-space of J is one-dimensional. It is obvious that the null space of J is of dimension at least 2 at a branch point. Thus, the following can be used to characterize and identify branch points

$$\text{Rank}(N(J)) > 1. \tag{9}$$

Further, the null vectors at a branch point can be used to locally characterize and follow the various branches of equlibria emanating from the point.

The stability of points on an equilibria curve can be found by carrying out a local eigenvalue analysis. The well known Gorbman Hartman theorem states that if all eigenvalues of the Jacobian matrix at a fixed point are in the open left half complex plain then the equilibrium is locally asymptotically stable. This result is used to characterize the stability of the fixed point curve. Bifurcations (or changes in stability) along the curve can be detected by observing the eigenvalues. A

computationally efficient way of finding bifurcation points is by the use of so-called *test functions*. Test functions for a bifurcation are functions that have a regular zero at the bifurcation point. For example, the determinant of the Jacobian matrix has a regular zero at a generic fold point, and can thus serve as a test function for a fold bifurcation.

2 Survey of Existing Tools

There exist several software packages to carry out bifurcation and continuation analysis of ODEs, iterated maps and PDEs. The aim of this section is to provide a overview of these tools, primarily to give the reader an idea of the state-of-art of the field. As one shall soon realize ABCT is indeed rudimentary compared to some of these advanced tools.

The following discussion on different tools is reproduced almost verbatim from Ref. [Dhooge et al., 2003].

1. AUTO86/97 [Doedel et al., 1998]. Runs on UNIX workstations under XWindows and has a rudimentary graphical user interface (GUI). Supports continuation of equilibrium and periodic and homoclinic solutions and their bifurcations. Computationally advanced but very difficult to use; does not provide any normal form analysis. The most recent version AUTO2000 is written in C and has a command language interface based on the scripting

language Python

2. DSTOOL [Back et al., 1992]. Runs on UNIX workstations under X-Windows, has a user-friendly tcl/tk-GUI and allows to visualize computed data with GEOMVIEW. Supports simulation and (limited) continuation analysis. Computes one-dimensional invariant manifolds of fixed points. Has an extensive documentation that is both developer- and user-oriented. Possible, but hard, to extend.

3. LOCBIF [Khibnik et al., 1992]. Runs as a DOS-application under MSWindows, has a simple GUI. Supports continuation of equilibrium and periodic solutions, as well as the normal form analysis of the simplest bifurcations of equilibria. Has many restrictions on the model complexity and a closed architecture.

4. CONTENT [Kuznetsov and Levitin, 1997]. Runs on UNIX workstations under X-Windows with (Open)Motif and on PCs underWindows-95/NT. Has a user-friendly GUI and an on-line hypertext help. Supports the continuation of equilibrium and periodic solutions and their bifurcations, as well as the normal form analysis of many bifurcations of equilibria and fixed points. Allows one to use a special linear algebra C-library for each solution type. Uses a very specific format to store the systems and results of computations. Automatically generates partial derivatives of the system right-hand sides.

Possible, but hard, to extend.

5. XPPAUT [Ermentrout, 2002]. Runs on UNIX workstations, Windows, and Mac OSX. Has a GUI with animation. Allows one to simulate ODEs, discontinuous differential equations, delay equations and differential algebraic equations, some BVPs and PDEs. Uses AUTO numerics for the continuation of equilibrium and periodic solutions and their bifurcations.

6. MATCONT [Dhooge et al., 2003]. Is an implementation and extension of CONTENT in the MATLAB environment. Has a GUI and an stiff learning curve. Provides access to all MATLAB utilities and can interface with many toolkits written for MATLAB.

7. PyCont. Is a sub-package in PyDSTOOLs, which is an implementation and extension of DSTOOLs in a python based environment. Easy to use and provides access to utilities of python computing environment NUMPY, and plotting utilities. Software is in developmental stage.

Table 1[1] summarizes the capabilities of the prominent software tools. Compared to these ABCT can only do the following: 1.) continuation of equilibria, 2.) detection of branch points and codim 1 bifurcations of equilibria and 3.) computation of normal form coefficients for codim 1 bifurcations of equilibria.

[1]taken almost verbatim from Dr. Kuznetsov's webpage `http://www.math.uu.nl/people/kuznet/res.html`

Table 1: Capabilities of standard bifurcation analysis tools. A=Auto, C=CONTENT, M=MATCONT, P=PyCont.

Capability	A	C	M	P
time-integration		+	+	+
Poincaré maps			+	
continuation of equilibria	+	+	+	+
detection of branch points and codim 1 bifurcations of equilibria	+	+	+	+
computation of normal forms for codim 1 bifurcations of equilibria		+	+	+
continuation of codim 1 bifurcations of equilibria	+	+	+	+
detection of codim 2 equilibrium bifurcations (cusp, Bogdanov-Takens, fold-Hopf, generalized and double Hopf)		+	+	+
continuation of limit cycles	+	+	+	+
detection of branch points and codim 1 bifurcations (limit points, flip and N-S) of cycles	+	+	+	+
continuation of codim 1 bifurcations of cycles	+		+	+
branch switching at equilibrium and cycle bifurcations	+	+	+	+
continuation of branching points of equilibria and cycles	+		+	+
computation of normal forms for codim 1 bifurcations of cycles			+	+
detection of codim 2 bifurcations of cycles			+	
continuation of orbits homoclinic to equilibria	+			

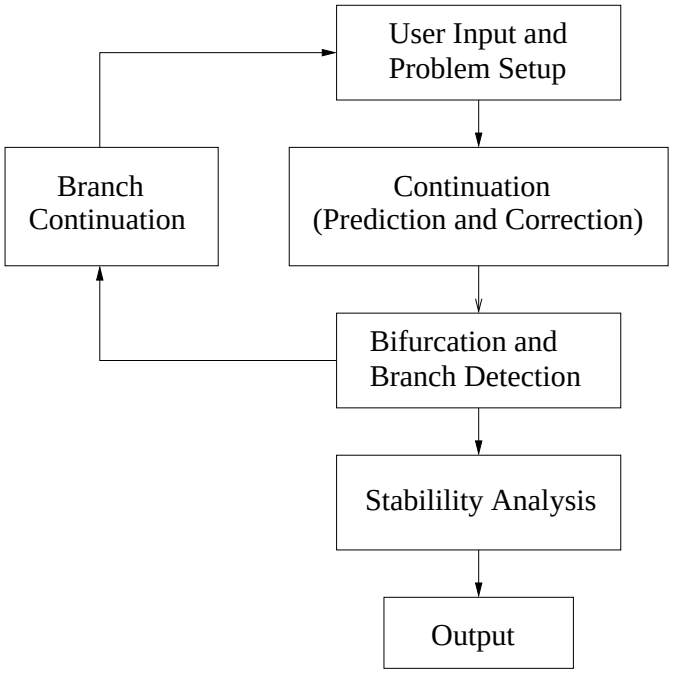

Figure 2: Data flow in ABCT

3 ABCT: Implementation and Data Flow

Figure 2 shows the data flow in a typical ABCT session. The user begins with defining the system and setting a continuation problem by specifying a fixed point, giving parameter limits etc. The code implements fixed step continuation using tangent prediction and pseudo arc-length based Newton correction. Each point on the curve is checked for stability, bifurcations and branch points. If a bifurcation or a branch point is encountered then a message is printed to the console. Based on this information the user can make changes to the problem definition to include continuation of the branches encountered. Optionally the user can also write the

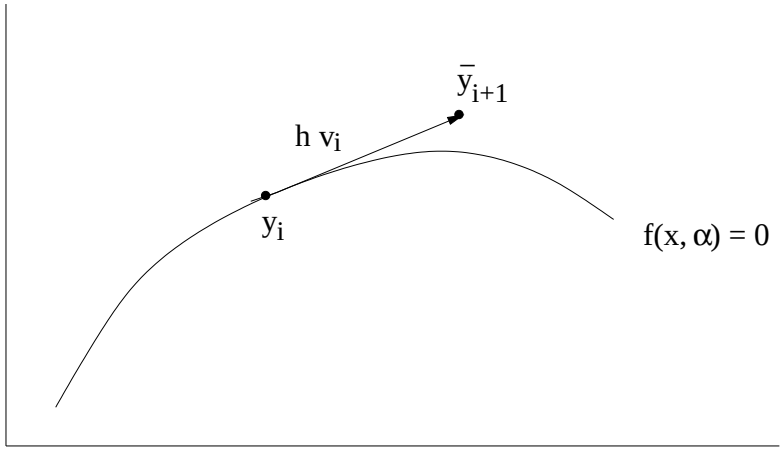

Figure 3: Tangent prediction

data to a file and draw bifurcation plots. A simple script is provided to plot the diagrams using Gnuplot [Williams and Kelley, 1998].

The algorithms used in ABCT are described in the following sections. A more complete account of these algorithms can be found in one of the following three references Kuznetsov [Kuznetsov, 2004], Allgower and Georg [Allgower and Georg, 2003], and Press [Press, 1992].

3.1 Prediction and Correction

Suppose that we are interested in continuing the implicit curve

$$f(x, \alpha) = 0, \tag{10}$$

from a regular point y_i. According to the tangent prediction, the predicted point $\bar{y}_{i+1}$ is

$$\bar{y}_{i+1} = hv_i + y_i, \tag{11}$$

where h is the step-size and v_i is the unit tangent vector defined by $J(y_i)v = 0$ (see figure 3). $J(y_i)$ is the Jacobian evaluated at the point y_i,

$$J(y_i) = \left.\frac{\partial f}{\partial y}\right|_{y=y_i}. \tag{12}$$

It should be noted that Eq. 7 defines v to be a unit vector. At a regular point Eqs. 5, 7 define the vector v up to a $\pm$ sign. The sign is fixed using the continuity of vector v along the curve

$$v_i \cdot v_{i-1} > 0, \tag{13}$$

where $(\cdot)$ is the standard vector inner product.

After finding a predicted point $\bar{y}_{i+1}$ it is necessary to correct the point to the required precision. This correction is achieved using Newton-Ralphson iteration like methods. Notice that $f(y) = 0$ gives only n equations where n is the dimensionality of the state vector. The vector y is $n + 1$ dimensional, thus one needs to specify one more condition before applying Newton-Ralphson like methods to the system. There are many ways to impose this additional condition. ABCT implements the pseudo-arclenght condition given by

$$g(y_{i+1}) \equiv (y_{i+1} - y_i) \cdot v_i - h_i = 0. \tag{14}$$

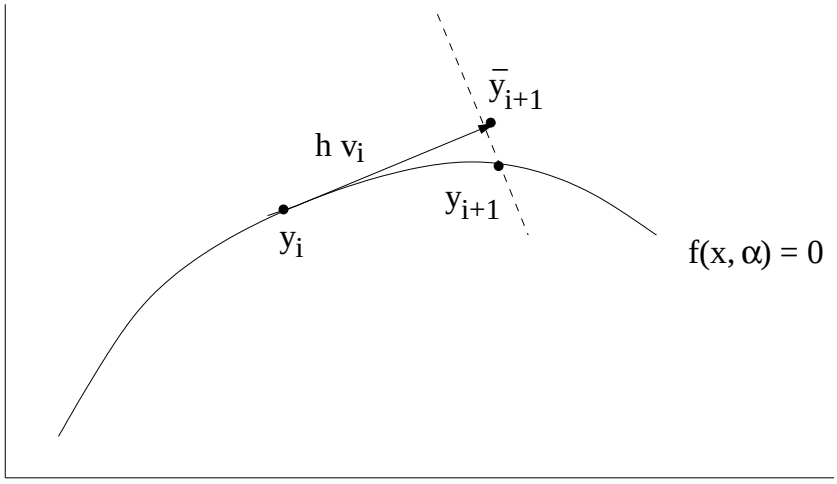

Figure 4: Pseudo-arclength correction

Thus, the augmented system iterated by the Newton-Ralphson like method is

$$\begin{aligned} f(y_{i+1}) &= 0, \\ g(y_{i+1}) &= 0. \end{aligned} \tag{15}$$

This prediction-correction scheme is often called the pseudo-arclength continuation and is shown graphically in figure 4.

3.2 Branch Point Detection

The appearance of a branch point along the continued curve is indicated by an increase in dimension of the null-space of the Jacobian matrix $J(y)$. Suppose y_n is a point where the matrix $J(y_n)$ has a higher dimensional null-space. Numerically any point y_i will be only an approximation to the point y_n. Thus, the question weather a point y_i is a branching point or not is not a numerically robust question.

We classify a point as a branching point if the ratio of the smallest singular value of $J(y_i)$ to its largest singular value is less than some prescribed number. If $J(y_i)$ has a two dimensional null-space then this number is zero. Once the dimension of the null-space is ascertained in this manner, the vectors corresponding to the small singular values are the basis vectors for the null space. The different branches can then be traced using each of these vectors in the tangent prediction described in the previous section.

3.3 Bifurcation Detection

At each point on the fixed point curve one can evaluate a certain test function and conclude about the presence of bifurcations by the behavior of this function and certain associated degeneracy conditions. These functions have regular zero at the bifurcation points. The test function and degeneracy conditions for fold and Hopf bifurcations are discussed here. A more complete discussion of these topics can be found in Kuznetsov [Kuznetsov, 2004].

3.3.1 Fold Points

The since the matrix

$$J' = \frac{\partial f}{\partial x} \tag{16}$$

has a single zero eigenvalue at a generic fold point, so the determinant of this matrix can be used as a test function for fold point detection.

$$\xi(y_i) = \det\left(\left.\frac{\partial f}{\partial x}\right|_{y=y_i}\right) \tag{17}$$

is a suitable test function for fold points. It can be proved that at a generic fold point the restriction of the system to its one-dimensional center-manifold can be written as

$$\dot{u} = bu^2 + O(u^3). \tag{18}$$

A generic fold bifurcation occurs if $b \neq 0$ and the transversality (non-zero speed crossing of eigenvalues) with respect to the parameter is maintained. The reader is referred to Kuznetsov [Kuznetsov, 2004] for details on calculating b.

3.3.2 Hopf points

A suitable test function for Hopf points can be found by invoking Stéphanos theorem. It is mentioned here that the following function can serve as a test function for Hopf points

$$\eta(y_i) = \det\left(2\frac{\partial f}{\partial x} \bullet I_n\right)\Bigg|_{y=y_i}, \tag{19}$$

where $(\bullet)$ is the bialternate product and I_n is the $n \times n$ identity matrix. The non-degeneracy conditions for the Hopf bifurcation are transversality with parameter and a non-zero first Lyapunov exponent. The bifurcation is super or

sub-critical depending on the sign of the exponent. The details can be found in Kuznetsov [Kuznetsov, 2004].

3.4 Stability Analysis

Eigenvalue analysis is used to characterize the stability analysis at points on the fixed point curve. A point y_i is stable if the matrix

$$J' = \left.\frac{\partial f}{\partial x}\right|_{y=y_i} \tag{20}$$

has all left half plane eigenvalues, and unstable if it has any right half plane eigenvalue. The points were the eigenvalues have zero real part correspond to bifurcation points and are dealt accordingly. Algorithms for finding eigenvalues of general matrices can be found in Ref. [Press, 1992].

4 Interface and Results

We demonstrate the use of the code by carrying out the bifurcation analysis for the following system

$$\dot{x} = \alpha x - x^3 \tag{21}$$

4.1 User Functions

The user has to program two function. First is called `ABCT_dyn.c`. This function has the information about the system dynamics and reads as follows

```
#include ‘‘ABCT.h’’


float * ABCT_dyn(float *fp, int dim)
// Return the RHS of x_dot = f(x,p) where fp = [x,p]
// dim = number of states
{
    float *f = (float *)ABCT_MemoryAlloc(dim,0,’f’);
    f[1] = -fp[1]*fp[1]*fp[1] + fp[2]*fp[1];


    return f;
}
```

The parameter α is in the $n+1$ index of the array `fp`. The other function programmed by the user is the main function that determines the program flow. This function is in a file named `ABCT.c`.

```
#include ‘‘ABCT.h’’


int main({


        // Set up parameters
        int dim = 1;    // 1-D system
```

```
int type = 0;   // Starting point is a regular fixed point

float *fp = (float *)ABCT_MemoryAlloc(dim+1,0,'f');

fp[1] = 0.0;            // fixed point x = 0

fp[2] = -1.0;           // alpha = -1.0


float *vl = (float *)ABCT_MemoryAlloc(dim+1,0,'f');


// upper and lower limits on alpha

float plow = -2.0;

float phigh = 2.0;


// step size

float ds = 0.01;


float **b = (float **)ABCT_MemoryAlloc(500,dim+2,'f');

float **bp1 = (float **)ABCT_MemoryAlloc(50,dim+2,'f');

float **nv1 = (float **)ABCT_MemoryAlloc(50,dim+2,'f');


int np = 0;

int nbp = 0;
```

```
FILE *pfs = fopen(``SBranch.txt'',``w+'');

FILE *pfus = fopen(``UBranch.txt'',``w+'');


// Continute First Branch

ABCT_ContinueBranch(fp, dim, type, vl, plow, phigh, ds, b, bp1, nv1, &np,


ABCT_WriteBifFile(dim, np, b, pfs, pfus);


// Continute Second Branch

// fixed point is the branch point

fp[1] = bp1[1][1];

fp[2] = bp1[1][2];


// tangent vector is the first null vector

vl[1] = nv1[1][1];

vl[2] = nv1[1][2];

type = 1;

float **bp2 = (float **)ABCT_MemoryAlloc(50,dim+2,'f');

float **nv2 = (float **)ABCT_MemoryAlloc(50,dim+2,'f');

ABCT_ContinueBranch(fp, dim, type, vl, plow, phigh, ds, b, bp2, nv2, &np,
```

```
        ABCT_WriteBifFile(dim, np, b, pfs,pfus);



        // Continute Third Branch
        // fixed point is the branch point
        fp[1] = bp1[1][1];
        fp[2] = bp1[1][2];

        // tangent vector is the first null vector
        vl[1] = nv1[1][1];
        vl[2] = nv1[1][2];
        // however, ds is set to -ds to continue in opposite direction
        ABCT_ContinueBranch(fp, dim, type, vl, plow, phigh, -ds, b, bp2, nv2, &np, &nbp)
        ABCT_WriteBifFile(dim, np, b, pfs,pfus);
}
```

Compiling and running the code results in a Postscript file containing the bifurcation diagram and file containing the data.

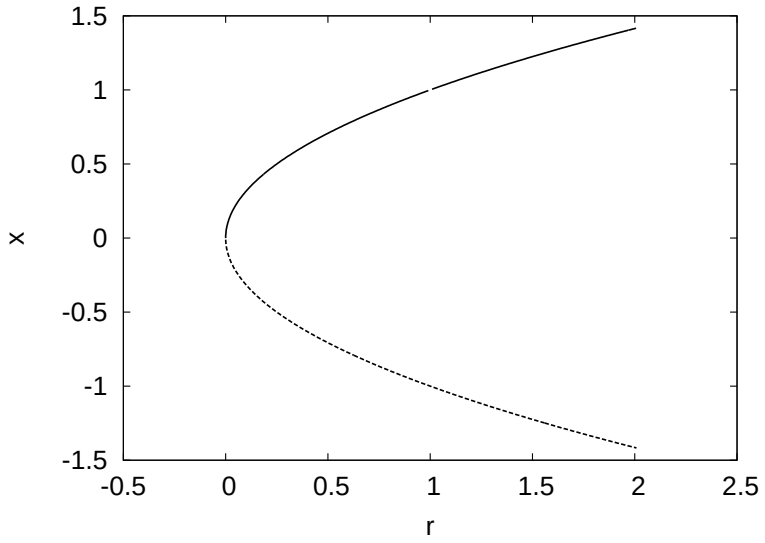

Figure 5: Typical fold point. $\dot{x} = -r + x^2$

4.2 Typical Results

In this section we present some typical bifurcation diagrams obtained with the code.

References

E. Allgower and K. Georg. Continuation and path following. *Acta Numerica 1993*, pages 1–64, 1993.

E. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, 2003.

A. Back, J. Guckenheimer, M. Myers, F. Wicklin, and P. Worfolk. DsTool: Com-

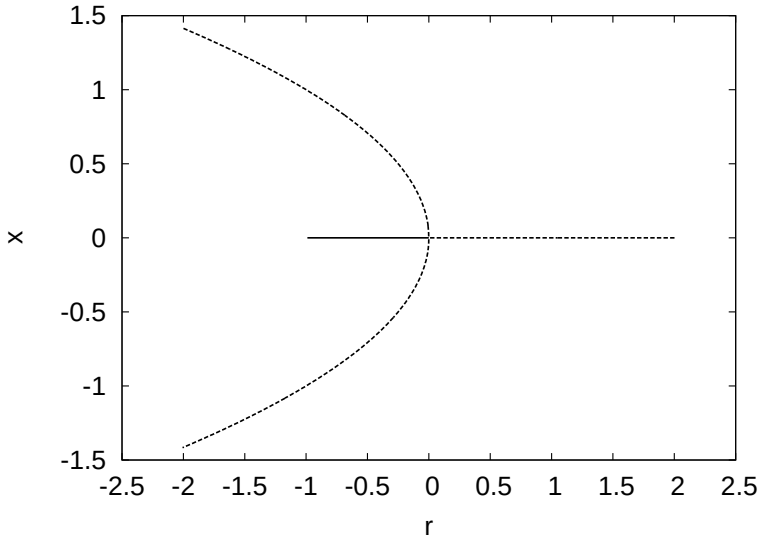

Figure 6: Sub-critical pitchfork. $\dot{x} = rx + x^3$

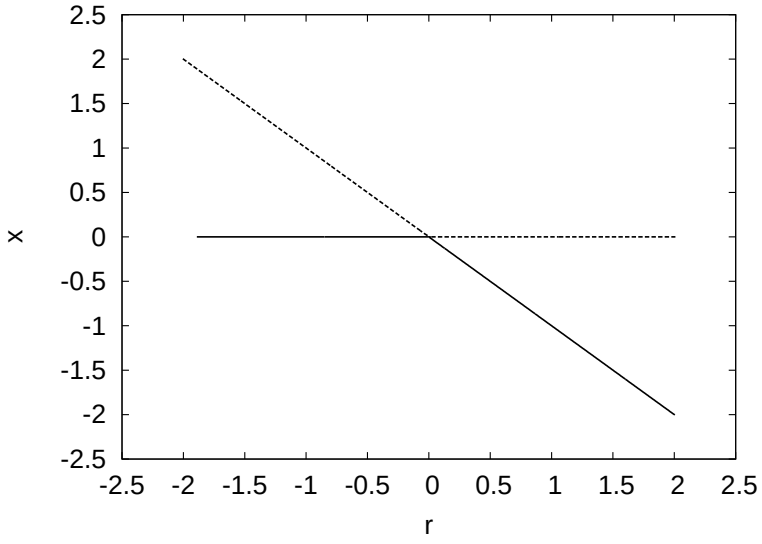

Figure 7: Transcritical. $\dot{x} = rx - x^2$

puter assisted exploration of dynamical systems. *Notices Amer. Math. Soc*, 39 (4):303–309, 1992.

A. Dhooge, W. Govaerts, and Y. Kuznetsov. MATCONT: A MATLAB package for numerical bifurcation analysis of ODEs. *ACM Transactions on Mathematical Software (TOMS)*, 29(2):141–164, 2003.

E. Doedel, H. Keller, and J. Kernevez. Numerical analysis and control of bifurcation problems. I. Bifurcation in finite dimensions. *Internat. J. Bifur. Chaos Appl. Sci. Engrg*, 1(3):493–520, 1991.

E. Doedel, A. Champneys, T. Fairgrieve, Y. Kuznetsov, B. Sandstede, and X. Wang. AUTO97: Continuation and bifurcation software for ordinary differential equations (with HomCont). *Concordia University, Montreal, Canada*, 1998.

B. Ermentrout. *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*. Society for Industrial Mathematics, 2002.

H. Keller. *Lectures on Numerical Methods in Bifurcation Problems*. Springer, 1987.

A. Khibnik, Y. Kuznetsov, V. Levitin, and E. Nikolaev. LOCBIF. *Interactive LOCal BIFfurcation Analyzer, preprint*, 1992.

Y. Kuznetsov. *Elements Of Applied Bifurcation Theory.* Springer, 2004.

Y. Kuznetsov and V. Levitin. CONTENT: A multiplatform environment for continuation and bifurcation analysis of dynamical systems. *Centrum voor Wiskunde en Informatica, Kruislaan*, 413:1098, 1997.

W. Press. *Numerical Recipes in C: the art of scientific computing.* Cambridge University Press, 1992.

T. Williams and C. Kelley. GNUPLOT: An Interactive Plotting Program. *Manual*, v. 3, 1998.

W. Zangwill and C. Garcia. *Pathways to solutions, fixed points, and equilibria.* Prentice-Hall Englewood Cliffs, NJ, 1981.